Ajuntament de Barcelona

Conceptes bàsics de programació

IT Academy

24 de febrer 2021

Barcelona Activa

Unió Europea
Fons Europeu
de Desenvolupament Regional

Índex

Variables

Operadors

Estructures condicionals

Estructures repetitives

Funcions

Variables: Definició

- Una **variable** és un **contenidor** d'un **valor.**
- Aquest valor pot ser d'un **tipus bàsic**
- Els tipus bàsics disponibles **depenen del llenguatge de programació en el que treballeu.**
- Generalment, les variables **es declaren**, és a dir, s'explicita en el programa la seva existència.
- Si el codi font d'un programa fos un text narratiu, les variables serien els **noms/substantius**.

Variables: Exemple en codi font

```
var a = 3;
```

```
int a = 3;
```

"a" és el **nom de la variable**, amb aquest nom ens referirem a la variable durant l'execució del programa

"3" seria el valor de la variable, que en aquest cas representa el nombre enter 3.

"var" i "**int**" son paraules clau(**keywords)** que serveixen per explicar al programa que, lo que va a continuació es la declaració d'una variable.

En alguns llenguatges caldrà especificar el tipus de variable (llenguatges fortament tipats) com al cas "int" on s'especifica que la variable a declarar és de tipus **enter** mentre que en d'altres llenguatges(dèbilment tipats) no caldrà especificar el tipus de la variable en la declaració.

Variables: Tipus bàsics

Com hem comentat abans, els tipus bàsics depenen del llenguatge que utilitzem, pero per lo general acostumen a ser:

- **Enters:** Per a valors numèrics enters
- **Floats/Doubles:** Per a valors numèrics amb decimals
- **Boolean:** Per a representar **true**(cert) o **false(**fals**)**
- **String:** Per representar cadenes de caracters. Exemple: “Hola”

Més endavant, amb aquests tipus bàsics podrem composar estructures de tipus més complexes(objectes)

Operadors lògics

Amb les variables podem fer diferents **operacions**, que **variaran en funció del tipus de variable amb que operem**.

- Arimètics
- Unaris
- Relacionals
- Asignació
- Lògics

Operadors aritmètics

S'utilitzen per realitzar operacions aritmètiques simples amb tipus bàsics de variables. Necessiten un mínim de dos variables.

- **Suma (+):** `var resultat = a + b;`
- **Resta (-):** `var resultat = a - b;`
- **Mòdul (%):** `var resultat = a % b;`
- **Multiplicació (*):** `var resultat = a * b;`
- **Divisió(/):** `var resultat = a / b;`

Operadors unaris

Operacions sobre una única variable.

- **Negació lògica (!):** En un valor boolea, inverteix el valor.(true a false o, false a true)

```
var boolea = true;
boolea = !boolea;        //boolea pasa a ser false
```

- **Increment(++):** Incrementa el valor d'una variable en 1.

```
var a = 3;
++a; // A passa a tenir valor 4
```

- **Decrement (--):** Decrementa el valor d'una variable en 1.

```
var a = 3;
--a; // A passa a tenir valor 2
```

Operadors relacionals

Comparen variables en base a una relació, tornant **true** si es compleix la relació, o **false** en cas contrari.

- **Igual que (==):** Torna **true** si les variables comparades tenen el **mateix valor.**
- **No igual que (!=):** Torna **true** si les variables comparades tenen **diferent valor.**
- **Menor que(<):** Torna **true** si la primera variable té **valor inferior** a la segona.
- **Menor o igual que (<=):** Torna **true** si la primera variable té **valor inferior o igual** a la segona.
- **Major o igual que (>):** Torna **true** si la primera variable té **valor superior** a la segona.
- **Major que(>=):** Torna **true** si la primera variable té **superior o igual** a la segona.

Operadors relacionals: Exemples

```
var x = 3;
var y = 3;

x == y;  // retorna true

x != y; // retorna false
```

```
var x = -2;
var y = 4;

x > y;  // retorna false

x < y;  // retorna true

x <= y; // retorna true
```

Operadors d'assignació

Varien el valor de la variable a la que s'assigna.

- **Assignació simple** (=): Assigna a la variable un valor.

- `var a = 5;`

- `var salutacio = "Hola món!";`

- **Declaració composta** : Quan combinem assignació de valor i operació aritmètica.

- `var a = 5;`
- `var a += 3; // a passa a valor 8.`
- `var a -= 2; // a passa a valor 6.`

Operadors lògics

Compara valors booleans per a permetre la creació de condicions més complexes

- **AND lògic** (**&&**): Retorna **true** si **totes** les condiciones disposades son **true**.
- **OR lògic(||)**: Retorna **true** si **al menys una** de les condiciones disposades és **true**.

```
var a = 3;
var b = 2;
var c = -1

(a > b) && ( b > c) // És true
(a > b) && ( b < c) // És false
(a  > b) || (b < c) // És true
(a  < b) || (b < c) // És false
```

Estructures condicionals

De tant en tant, els programes necesiten fer **preguntes directes**(que retornin **true** o **false**, per tal de decidir executar un o altre grup d'instruccions.

És aquí on entren les diferents maneres de fer-ho, típicament:

- **if-else-else if**
- **switch**

Estructures condicionals: if-else-if

Els parèntesis serveixen per separar la condició de la resta de sentències del programa. Una condició pot ser qualsevol expressió que retorni un valor **booleà (true o false)**

```
if(condicio) {
     //Si la condició és certa, s'executarà aquest codi font
}
else if(condicio) {
     //Si aquesta és la  condició certa, s'executarà aquest codi font
}
else {
     //Si la condició és falsa s'executarà aquest codi font
}
```

Entre corxets posem les línies de codi font que volem que s'executin en cas de que la condició sigui true.

Estructures condicionals: if-else-if

Exemple:

```
var a = 4;

var b = 3;


if(a > b) {
     ++b;
}
else if(a < b) {
     ++a;
}
else {
     //Aquí només entraria si a i b tinguessin el mateix valor
}
```

En aquesta lògica, s'executaria la sentència "++b" donat que el valor de a és més gran que el valor de b

Estructures condicionals: switch

Una altra manera de “fer preguntes” en un programa és mitjançant l’estructura condicional **switch**

```
switch(expressio) {

     case valor1:
          //codi font
          break;
     case valor2:
          //codi font
          break;

     default:
          //codi font
          break;

}
```

En aquesta estructura s’especifiquen els valors que hauria de tornat l’expressió entre parèntesis, que, hauria de tornar un valor relacionat amb els casos a tenir amb compte

Important el **break** al final de cada cas, o s’executaràn tots els casos seguits !

Estructures condicionals: switch

Exemple:

```
switch(expressio) {

     case 'Java':
          //codi font
          break;
     case 'Javascript':
          //codi font
          break;

     default:
          //codi font
          break;

}
```

Si la variable "**expressio**" conté com a valor la paraula "Java" o "Javascript", s'executaran les lògiques relacionades.

En cas contrari, s'executarà el codi font del cas **default**

Estructures repetitives

A vegades convé l'execució d'un mateix grup de líneas de codi font de forma reiterada, i és aquí on trobem les instruccions de repetició.

- **while**
- **for**
- **do while**

Estructures repetitives: while

```
while(condicio) {
    //execució de codi font
}
```

Mentre es compleixi la condició, s'executarà el codi font entre corxets. De nou, en aquesta cas la condició ha retornar un valor booleà

Exemple:

```
var a = 0;
while(a < 5) {
    //execució de codi font
    ++a;
}
```

El codi font entre corxets s'executarà **5 vegades** en aquest cas.

IMPORTANT: Hem de tenir sempre en compte la modificació del valor(en aquest cas "++a") que s'evalua a la condició del bucle donat que sino es produeix mai la condició que trenqui el bucle, tindrem un **bucle infinit** que és font típica d'errors

Estructures repetitives: do while

```
do {
    //execució de codi font
} while(condicio)
```

Mentre es compleixi la condició, s'executarà el codi font entre corxets. De nou, en aquesta cas la condició ha retornar un valor booleà, només que aquesta vegada, la condició s'evalua **després** de l'execució del codi font.

Exemple:

```
var a = 0;
do {
    //execució de codi font
    ++a;
} while(a < 5)
```

El codi font entre corxets s'executarà **5 vegades** en aquest cas.

Estructures repetitives: for

```
for(var i = 0; condició; ++i)
{
    //codi font
}
```

El bucle **for** pot declarar en una mateixa linea de codi font, la **variable contadora**, la **condició** que evalua si el bucle continua o no i, la **variació del valor** que s'utilitzar per evaluar la condició per continuar iterant.

Exemple:

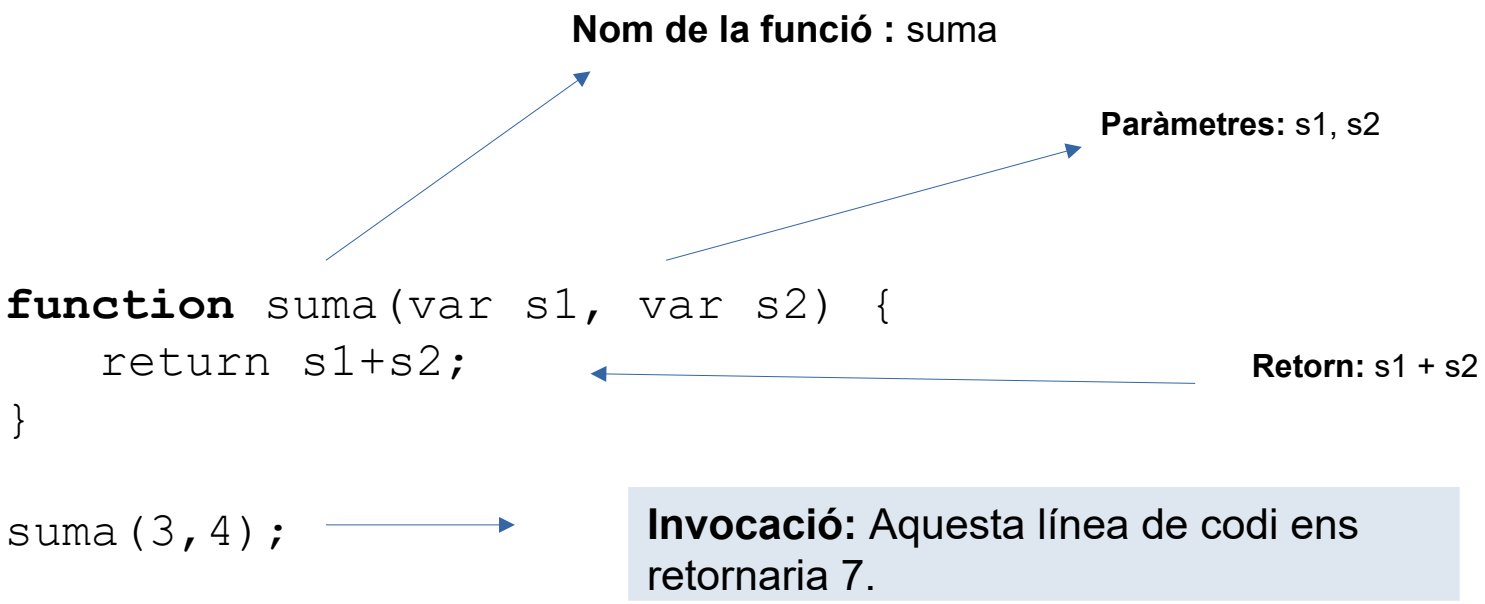
```
for(var i = 0; i < 7; ++i) {
    //codi font
}
```

El codi font entre corxets s'executarà 7 **vegades** en aquest cas.

Funcions: Definició

- Són **conjunts de líneas de codi font** que s'executen totes secuencialment mitjançant **invocació**.
- Si en lloc de codi font estiguéssim escribint un llibre, podriem dir que les funcions son **els verbs**.
- Són una altra peça bàsica en el mòn de la programació i ens ajuden a organitzar i entendre millor el codi font.

Funcions: Parts



Aqui es mostra un exemple bàsic, però imagineu això amb lògiques més complexes, que puguin ser invocades en una sola línea de codi font...

Conclusions

Aquest ha estat un senzill recorregut per conceptes comuns a una gran majoria dels llenguatges de programació orientats a objectes, dels quals aprendreu al menys un durant l'itinerari que trieu.

Tot i que aquests conceptes són comuns, cada llenguatge pot tenir les seves particularitats alhora de traslladar-ho al codi font, no us preocupeu, això se us explicarà quan arribeu ;)